Data-Oriented Parsing

Rens Bod, Remko Scha and Khalil Sima'an (editors)

August 8, 2001

CENTER FOR THE STUDY
OF LANGUAGE
AND INFORMATION

2

Parsing DOP with Monte-Carlo Techniques

Jean-Cédric Chappelier, Martin Rajman

2.1 Motivations

As shown by Sima'an (1996), finding, for a given sentence, the most probable parse tree (MPP) according to a Stochastic Tree Substitution Grammar (STSG) (Schabes 1992, Bod 1998), such as, for instance, a DOP grammar, appears to be an NP-hard problem and, as such, cannot be solved exactly in an effective way. Heuristics and/or approximations need to be used instead. Monte-Carlo sampling is one example of such an approach: it approximates the exact NP-hard maximization problem by a search over a reduced random sample that can be generated at reasonable (i.e. polynomial) algorithmic cost (Bod 1992).

The purpose of the present chapter is to provide theoretical justification for the use of Monte-Carlo sampling for finding the MPP in the framework of Data-Oriented Parsing; and to show how such sampling techniques can be efficiently implemented and, most of all, statistically controlled, so as to guarantee, with some *a priori* fixed probability error, that an obtained approximate solution indeed corresponds to an exact solution of the problem.

Section 2.2 provides a short introduction to the general problem of finding the MPP within the DOP framework. In sections 2.3 and 2.4, two generic implementations for an approximated search for the MPP based on Monte-Carlo sampling are presented (rescored sampling and exact sampling). The issue of the statistical control of the probability error during sampling is addressed in section 2.6. Finally, experimental

Data-Oriented Parsing.
Rens Bod, Remko Scha and Khalil Sima'an (editors).

results are presented, followed by some conclusions.

2.2 General Framework for the MPP in DOP

2.2.1 The DOP Model

As detailed in chapter ??, DOP constitutes a particular application of the Stochastic Tree Substitution Grammar (STSG) formalism. A STSG $\langle V_N, V_T, S, R, P\rangle$ (using the same notations as in chapter ??) is a grammar, the productions (or "rules") of which consist of elementary trees, that can be combined with the substitution operator to produce derivations of complete parse trees.

Each elementary tree t is assigned a probability $P(t)$ proportional to its number of occurrences in a reference treebank. The probability $P(d)$ of a derivation $d = \langle t_1, ..., t_q\rangle$ is then defined as

$$P(d) = \prod_{t\in d} P(t) = \prod_{i=1}^{q} P(t_i)$$

where $t \in d$ means "for all the occurrences of elementary trees appearing in the derivation d". Finally, the DOP-probability of a parse tree T is defined as the sum of the probabilities of all its derivations:

$$P_{\text{DOP}}(T) = \sum_{d\Rightarrow T} P(d) = \sum_{d\Rightarrow T} \prod_{t\in d} P(t)$$

where the subscript "$d \Rightarrow T$" means "for all derivations d of the parse tree T".

Notice that we need to distinguish between $P_{\text{DOP}}(T)$ and $P(T)$, as a given tree T might be a parse tree and an elementary tree at the same time. In such a case, these two probabilities might be different.

For a given sentence $w_1^m = w_1...w_m$ we also define

$$P_{\text{DOP}}(w_1^m) = \sum_{T\Rightarrow^* w_1^m} P_{\text{DOP}}(T) = \sum_{d\in\left(S\Rightarrow^* w_1^m\right)} P(d)$$

where "$T \Rightarrow^* w_1^m$" means that "T is a possible parse tree of w_1^m" and "$d \in (S \Rightarrow^* w_1^m)$" means "for all possible derivations d of w_1^m starting from S (i.e. $S \Rightarrow^* w_1^m$)".

In addition, for a tree T, $root(T)$ will denote its root, $frontier(T)$ its frontier, i.e. the ordered sequence of its leaves, $l_i(T)$ the ith leaf in $frontier(T)$, $\text{Rl}(T)$ the right-most leaf and $\text{Lls}(T)$ the sequence of all the leaves of T but the right-most one.

2.2.2 Parsing and Most Probable Parse Extraction

Parsing usually consists of two distinct phases:

analysis, during which some representation of the set of the derivations of the input string is built;

extraction, during which specific results are derived from the former representation; e.g. displaying all the parse trees, extracting the MPP, etc...

In the STSGs case, analysis can be achieved in cubic time (with respect to the input size), as it is the case for Context-Free Grammars (CFG). However, extracting the MPP from the resulting compact representation is an NP-hard problem (Sima'an 1996). This means that finding the MPP for an input sentence cannot (in general) be achieved in polynomial time[1], and heuristics and/or approximations need to be used instead. Monte-Carlo sampling is an example of an approach where the exact search for the MPP is approximated by a search over a reduced random sample of derivations that can be generated at polynomial algorithmic cost (Bod 1992).

The purpose of the following sections is to present two of such approximations, namely rescored and exact sampling. The next section first provides a brief description of the bottom-up item-based chart parsing technique we use for the implementation.

2.2.3 Parsing Analysis Phase

In item-based chart parsing, the analysis phase consists in filling up a chart with items that represent, in a compact form, all the possible derivations of all the possible substrings of the input sentence w_1^m. More precisely, the chart contains two different kinds of items:

- items of the form $[X, i, j]$ (for some $X \in V = V_N \cup V_T$, and $1 \leq i \leq j \leq m$) representing the fact that the substring $w_i^j = w_i ... w_j$ of the input sentence can be derived from X (this will be denoted by $X \Rightarrow^* w_i^j$);
- and items of the form $[\alpha\bullet, i, j]$ ($\alpha \in V^+$) representing the fact that the sequence of symbols α derives w_i^j and that α is a left-most sequence of leaves of some elementary tree; more formally, that

$$\alpha \Rightarrow^* w_i^j \text{ and } \left(\exists \beta \in V^+ \; \exists t \in R \text{ s.t. } \mathit{frontier}(t) = \alpha\beta\right)$$

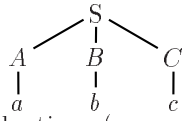
For a given symbol X ($X \in V$), notice the difference between $[X, i, j]$ and $[X\bullet, i, j]$: in both cases, X derives the substring w_i^j, but for the

[1] Notice however that finding the most probable derivation (MPD) can still be achieved in polynomial time with the standard algorithms used for Stochastic CFG (Jelinek et al. 1992, Stolcke 1995, Chappelier and Rajman 1998). Similarly, if well chosen constraints are imposed on the elementary tree set, MPP search can also be implemented in polynomial time (Chappelier and Rajman 2001).

second item to be produced, there must furthermore be a rule in the grammar with X as its the first right-hand side non-terminal.[2]

Items are produced ("derived") during the analysis phase either by combining items of the form $[\alpha\bullet, i, j]$ with items of the form $[X, j+1, k]$ or, in the case of unary productions, directly from items of the latter form. For example, the derivation

$$S \Rightarrow ABC \Rightarrow aBC \Rightarrow abC \Rightarrow abc$$

leading to the parse tree  will correspond to the following sequence of item productions ($\rightsquigarrow$ representing the production of an item): $[a,1,1] \rightsquigarrow [A,1,1]$, $[b,2,2] \rightsquigarrow [B,2,2]$,$[c,3,3] \rightsquigarrow [C,3,3]$,$[A,1,1] \rightsquigarrow [A\bullet,1,1]$,$[A\bullet,1,1] \oplus [B,2,2] \rightsquigarrow [AB\bullet,1,2]$ and $[AB\bullet,1,2] \oplus [C,3,3] \rightsquigarrow [S,1,3]$.

For the top-down sampling procedure that is required during the extraction phase and that will be detailed hereafter, we furthermore need to introduce the notion of *item decomposition*.

The decomposition of an item of the form $[A,i,j]$ ($A \in V_N$) is a pair $\prec t, k \succ_{[A,i,j]}$ consisting of:[3]

1. an elementary tree t that is the first step of some derivation d of w_i^j starting from A,
2. and a split value k ($i \leq k < j$) identifying the subsequence w_{k+1}^j derived by the right-most leaf of t in the derivation d.

When t is terminal (i.e. $frontier(t) = w_i^j$), the convention $k = 0$ will be used. Notice that, when t is not terminal, we have $\mathrm{Lls}(t) \Rightarrow^* w_i^k$ and $\mathrm{Rl}(t) \Rightarrow^* w_{k+1}^j$.

For items of the form $[\alpha\bullet, i, j]$, a decomposition simply reduces to the split value k, which indicates that α can be split into $\alpha = \beta X$ ($\beta \in V^+$, $X \in V$) such that $\beta \Rightarrow^* w_i^k$ and $X \Rightarrow^* w_{k+1}^j$. The convention $k = 0$ will be used for the special case $[X,i,j] \rightsquigarrow [X\bullet,i,j]$.

Notice that "terminal" items (i.e. items of the form $[a,i,j]$ or $[a\bullet,i,j]$, $a \in V_T$) do not have any decomposition.

To further simplify the notations, for any decomposition $\xi = \prec t, k \succ$, $P(\xi)$ will be used as another possible notation for the probability $P(t)$. By convention, for decompositions of the form $\prec k \succ$, $P(\prec k \succ)$ will be 1.

[2] Items of the second kind ($[\alpha\bullet, i, j]$) are essentially useful for parsing non-binary grammars, i.e. for SCFG, grammars that are not in Chomsky Normal Form and, for STSG, tree grammars containing non binary (nor unary) elementary trees.

[3] When not ambiguous, subscripts will be omitted for the sake of simplicity.

Finally, for any item I, $\mathcal{D}(I)$ will denote the set of all possible decompositions of I, and for any decomposition ξ of I ($\xi \in \mathcal{D}(I)$), when not ambiguous, $\mathcal{D}(\xi)$ will also be used as another possible notation for $\mathcal{D}(I)$.

Example

In order to illustrate the notions introduced so far, let us consider the very simple example of the parsing the input sequence "a a a" with the following CFG:

$$\texttt{S} \rightarrow \texttt{S S} \qquad\qquad \texttt{S} \rightarrow \texttt{a}$$

where a is the only terminal and S the only non-terminal of the grammar. In this case, the produced chart contains 9 items[4] with the following associated decomposition sets:

item	number of decompositions	decomposition sets
$[S,1,1]$	1	$\prec \texttt{S} \rightarrow \texttt{a}, 0 \succ$
$[S\bullet,1,1]$	1	$\prec 0 \succ$
$[S,2,2]$	1	$\prec \texttt{S} \rightarrow \texttt{a}, 0 \succ$
$[S\bullet,2,2]$	1	$\prec 0 \succ$
$[S,3,3]$	1	$\prec \texttt{S} \rightarrow \texttt{a}, 0 \succ$
$[S,1,2]$	1	$\prec \texttt{S} \rightarrow \texttt{S S}, 1 \succ$
$[S\bullet,1,2]$	1	$\prec 0 \succ$
$[S,2,3]$	1	$\prec \texttt{S} \rightarrow \texttt{S S}, 2 \succ$
$[S,1,3]$	2	$\prec \texttt{S} \rightarrow \texttt{S S}, 1 \succ$, $\prec \texttt{S} \rightarrow \texttt{S S}, 2 \succ$

For the item $I = [S,1,3]$ we have for instance:

$$\begin{aligned} \mathcal{D}(I) &= \mathcal{D}(\prec \texttt{S} \rightarrow \texttt{S S}, 1 \succ_I) = \mathcal{D}(\prec \texttt{S} \rightarrow \texttt{S S}, 2 \succ_I) \\ &= \{\prec \texttt{S} \rightarrow \texttt{S S}, 1 \succ_I, \prec \texttt{S} \rightarrow \texttt{S S}, 2 \succ_I\} \end{aligned}$$

2.2.4 Monte-Carlo Methods

Numerical methods known as Monte-Carlo methods (Hammersley and Handscomb 1964, Green 1995, Robert and Casella 1999) can loosely be described as statistical simulation methods. The expression "Monte-Carlo method" is actually very general. The name was inspired by Metropolis (1948) after the city in Monaco principality, because of the similarity between gambling games and the statistical sampling simulation done for the Manhattan Project during World War 2.

Monte-Carlo methods provide approximate solutions to a variety of mathematical problems by performing statistical sampling experiments

[4] Terminal items (i.e. items of the form $[a,i,i]$ in this case) as well as items of the form $[\alpha\bullet,i,3]$ (which cannot serve for the production of new items) have not been represented.

on a computer. Such methods can even apply to problems with no natural probabilistic content. For instance, Monte-Carlo methods can be used for the computation of the decimals of π (Hall 1873) or of some high-dimension integrals.

The only requirement is that the problem to be solved can be described in terms of a probability density function (PDF) on some configuration space, even if such a description may seem artificial. The Monte-Carlo method then proceeds by random sampling according to the PDF describing the problem. Many samplings are performed and the desired result is estimated on the the obtained samples. This requires that configurations can actually be generated according to the desired PDF and that the estimator can be computed on the samples and converges to the desired result when the number of samples grows to infinity.

For tractable problems, Monte-Carlo methods are among the least efficient numerical methods and shouldn't be used. However, for those intractable problems relying on N-point evaluations in a M dimensional space, the other numerical methods are often less efficient. In such cases indeed, as a consequence of the central limit theorem, Monte-Carlo methods have an absolute error estimation that decreases with $N^{-\frac{1}{2}}$ while most of the other methods usually decrease at best with $N^{-\frac{1}{M}}$. This property is one of the main reasons of the use of Monte-Carlo methods in our context.

2.2.5 Monte-Carlo Estimation of the MPP in DOP

In the framework of DOP, the purpose of Monte-Carlo estimation is to approximate the MPP extraction by a search over a limited random sample of derivations (Bod 1992).

The general underlying idea is the following: if a random set of derivations is produced with a known sampling distribution, the proportion in that set of derivations corresponding to a given parse tree will converge (due to the law of large numbers) to the sum of the sampling probabilities of all the possible derivations of this tree. This convergence property can then be used to estimate the DOP-probability of the parse tree if the following conditions are verified:

1. the sampling of derivations can be achieved in an efficient (i.e. polynomial) way;
2. the DOP-probability of any parse tree can be estimated from the sampled set of derivations; i.e. there exists a function that can be computed on the sampled set and indeed converges to DOP-probability when the size of the sampled set grows to infinity;
3. the estimation function can be computed on the sampled set in an

efficient way;

4. for the MPP, the convergence occurs (with some a priori controlled error) for a sample size significantly smaller than the total number of derivations.

We now present two sampling methods that both verify the three first above conditions. The problem of the control of the sampling will be addressed later, in section 2.4. For both methods, the general sampling algorithm described in table 1 is used. The difference will come from the specific implementation of the action "*choose a decomposition at random*".

From the compact representation of the derivation set produced during the analysis phase of the parsing, the sampling algorithm produces one derivation at a time by recursively choosing at random decompositions of items as described in table 1, starting from the top level item $[S, 1, m]$.

As any derivation of a string of size m is composed of $\mathcal{O}(m)$ decompositions[5], and choosing a decomposition of an item may be $\mathcal{O}(m)$ if not carefully implemented (see also (Goodman 1998) page 177), the proposed sampling algorithm might be of cost $\mathcal{O}(m^2)$. It can however easily be improved to $\mathcal{O}(m)$ if the storage of an item in the chart contains explicit pointers to its decomposition, since, in that case, choosing a decomposition of an item can be implemented in $\mathcal{O}(1)$ with respect to m.

In any case, the first of the four conditions previously mentioned is fulfilled. The main problem of Monte-Carlo approach is therefore not the sampling algorithm itself, but rather the choice for the implementation of the action "*choose a decomposition at random*", so that conditions 2 and 3 previously mentioned are fulfilled.

The two following sections detail two different solutions leading to a correct sampling for the items:

rescored sampling: the sampling probability used for randomly choosing the item decompositions in the chart is simple and *a priori* defined. In this case however, nothing guarantees that the limit towards which the proportion, in the sample, of derivations associated with a given parse tree converges is its DOP-probability. The relative frequencies of the derivations therefore need to be rescored *a posteriori*, so as to guarantee a correct convergence value for their sum;

exact sampling: the sampling probability used for the item decompo-

[5] If the grammar does not contain any cycle, which is the case we consider.

Sample$([X,i,j])$:

choose at random a decomposition $\xi = \prec t, k \succ$ of $[X,i,j]$
if $k = 0$ (unary productions)
if ξ is terminal (i.e $t = X \to w_i^j$)
return t
else
(here $t = X \to Y$, $Y \in V_N$)
return t, **Sample**$([Y,i,j])$
else
return t, **Sample**$([\mathrm{Lls}(t)\bullet, i, k])$, **Sample**$([\mathrm{Rl}(t), k+1, j])$

Sample$([\alpha\bullet,i,j])$:

choose at random a decomposition $\xi = \prec k \succ$ of $[\alpha\bullet,i,j]$
if $k = 0$ (i.e. $\alpha = X \in V$)
if $X \in V_T$
return
else (then $X \in V_N$)
return **Sample**$([X,i,j])$
else
(here $\alpha = \beta X$, $\beta \in V^+$, $X \in V$)
return **Sample**$([\beta\bullet,i,k])$, **Sample**$([X,k+1,j])$

TABLE 1 The general sampling algorithm: to produce at random a derivation of an input string w_1^m for which the item-based parse chart has already been generated, call **Sample**$([S,1,m])$, where S is the top-level symbol of the grammar.

sitions is specifically computed in such a way that the sampling probability of a parse tree (i.e. the sum of the sampling probabilities of all its derivations) is exactly equal to its (conditional) DOP-probability (knowing the input sentence).

The main advantage of the first approach is its simplicity and generality, while the advantage of the second one is a faster convergence, as well as the possibility of an *a priori* control over the error probability (and hence over the accuracy of the method). More precisely, the sampling can be statistically controlled by setting *a priori* an arbitrarily small probability (the error probability) corresponding to an upper bound for the probability that the approximated solution does not correspond to the exact solution.

2.3 Rescored Sampling

2.3.1 General Method

Let $d_i, i = 1...N$, be all the possible derivations of all the parse trees associated with a given input sentence. A random sampled set of size n is built by successively and independently applying n times the sampling algorithm described in the previous section. Let n_i be the number of occurrences of derivation d_i in the sampled set[6].

The empirical score $W^{(n)}(T)$ of a parse tree T is then defined by: $W^{(n)}(T) = \sum_{d_i \in T} \frac{n_i}{n} W_i$, where W_i is some rescoring factor for the derivation d_i. The whole interest of such a definition lies in the fact that, by the law of large numbers (i.e. when n grows to infinity), the empirical score $W^{(n)}(T)$ converges to $W^{(\infty)}(T) = \sum_{d_i \in T} P_{si} W_i$, where P_{si} the sampling probability of derivation d_i.

This convergence property establishes the link between the local random choices for derivations (expressed by the P_{si}) and the global scores obtained for the corresponding parse trees (corresponding to the $W^{(\infty)}(T)$). Provided that it is possible to compute both W_i and P_{si} for each sampled derivation d_i, it is then possible to estimate $W^{(\infty)}(T)$ by the empirical value $W^{(n)}(T)$. This is, of course, particularly useful if $W^{(\infty)}(T) = P_{\text{DOP}}(T)$, the original score to be maximized.

A first sampling scheme, called "*naive* sampling", consists in using a uniform sampling distribution for each of the independent random choices among all the decompositions of each of the items successively selected. By construction, this corresponds to $P_{si} = \prod_{\xi \in d_i} \frac{1}{|\mathcal{D}(\xi)|}$.[7] Without any rescoring (i.e. with $W_i = 1$), naive sampling leads, for every parse

[6] $\sum_{i=1}^{N} n_i = n$. Notice that some n_i may be null.

[7] The notation $\xi \in d_i$ used as a subscript of a sum or a product means "*for all item decompositions ξ chosen during the sampling of derivation d_i*".

tree T, to a score $W^{(\infty)}(T)$ of the form $\sum_{d_i \in T} \prod_{\xi \in d_i} \frac{1}{|\mathcal{D}(\xi)|}$, which is not the DOP-probability $P_{\text{DOP}}(T)$ to be estimated. Naive sampling can however be corrected, introducing rescoring factors $W_i = \frac{P(d_i)}{\prod_{\xi \in d_i} \frac{1}{|\mathcal{D}(\xi)|}} = P(d_i) \prod_{\xi \in d_i} |\mathcal{D}(\xi)|$, which leads to $W^{(\infty)}(T) = P_{\text{DOP}}(T)$. It is important to notice that these rescoring factors are easily computable during sampling. Indeed, as

$$W_i = P(d_i) \prod_{\xi \in d_i} |\mathcal{D}(\xi)| = \left(\prod_{\xi \in d_i} P(\xi) \right) \left(\prod_{\xi \in d_i} |\mathcal{D}(\xi)| \right) = \prod_{\xi \in d_i} \Big(P(\xi) \cdot |\mathcal{D}(\xi)| \Big)$$

W_i can be computed during the top-down extraction of each sampled derivation by cumulative multiplication of the $P(\xi) \cdot |\mathcal{D}(\xi)|$ values. Rescored naive sampling may therefore be implemented, providing a first very easy estimation method for the MPP search.

2.3.2 Existing Sampling Method for DOP

As Bod's original DOP sampling method (Bod 1995, Bod 1998) has been shown to suffer from several shortcomings (Goodman 1998) (page 179) (Hoogweg 2000) (section 6.2) (Chappelier and Rajman 2000), we here focus on its improved version (Hoogweg 2000). This sampling technique is exactly the one described above but with another sampling probability distribution. Indeed, for the local random decomposition choices, Hoogweg chooses "*directly the probabilities of the elementary subtrees*", the corresponding probability mass being *"the sum over the probabilities of all elementary trees that span the same subsentence and have the same root label*".[8] This means that the sampling probability of a decomposition ξ of $[X, i, j]$ is $P_s(\xi) = \frac{P(\xi)}{\sum_{\xi' \in \mathcal{D}(\xi)} P(\xi')}$, leading therefore, for any derivation d_i, to a $P_{s\,i}$ of the form $P_{s\,i} = \prod_{\xi \in d_i} \frac{P(\xi)}{\sum_{\xi' \in \mathcal{D}(\xi)} P(\xi')} = P(d_i) \cdot \frac{1}{\prod_{\xi \in d_i} \sum_{\xi' \in \mathcal{D}(\xi)} P(\xi')}$. The resulting score for a parse tree T is therefore $\sum_{d \in T} \frac{P(d)}{\prod_{\xi \in d} \sum_{\xi' \in \mathcal{D}(\xi)} P(\xi')}$ which, in the general case, is not the DOP-probability of that tree. To correct this behavior, the sampled derivations are rescored by $W_i = \prod_{\xi \in d_i} \sum_{\xi' \in \mathcal{D}(\xi)} P(\xi')$, which is computed during the sampling.[9]

Table 2 resumes the rescoring factors for the sampling methods seen so far.

[8] "*Elementary trees that span a given subsentence*" correspond to decompositions in our terminology.

[9] W_i is $\prod_j PM(t_j)$ with Hoogweg's notations.

method	$P_s(\xi)$	rescoring factor W_i
Hoogweg (2000)	$\frac{P(\xi)}{\sum_{\xi' \in \mathcal{D}(\xi)} P(\xi')}$	$\prod_{\xi \in d_i} \sum_{\xi' \in \mathcal{D}(\xi)} P(\xi')$
naive	$\frac{1}{\lvert\mathcal{D}(\xi)\rvert}$	$P(d_i) \cdot \prod_{\xi \in d_i} \lvert\mathcal{D}(\xi)\rvert$

TABLE 2 Rescoring factors for rescored sampling methods.

2.4 Exact Sampling

The purpose of exact sampling techniques is to ensure that the sampling probability of each parse tree is directly equal (i.e. without any rescoring) to its conditional DOP-probability knowing the input sentence, $P_{\text{DOP}}(T|w_1^m)$. Such sampling techniques therefore guarantee that the best parse tree also has the best sampling probability. Therefore, due to the law of large numbers (the sampling frequency converging towards the sampling probability), the most frequent tree in a sampled set has a good probability to be the MPP. One important additional advantage of exact sampling is the fact that it enables statistical control of the number of samples.

2.4.1 Computation of Sampling Probability for Decompositions

In this section, we derive the exact formula that has to be used for the sampling probability of the item decompositions so that the final sampling probability of a parse tree exactly corresponds to its DOP-probability conditional to the input sentence.

We first introduce an intermediate probability P_0 that will be useful for the computation. It is defined, for each item decomposition ξ, by:

- if ξ is of the form $\prec t, k \neq 0 \succ_{[root(t),i,j]}$:
$$P_0(\xi) = P(t) \cdot P_\Sigma([\text{Lls}(t)\bullet, i, k]) \cdot P_\Sigma([\text{Rl}(t), k+1, j])$$
- if ξ is of the form $\prec t, 0 \succ_{[root(t),i,j]}$ (i.e. ξ is terminal):
$$P_0(\xi) = P(\xi) = P(t)$$
- if ξ is of the form $\prec k \neq 0 \succ_{[\beta X\bullet,i,j]}$:
$$P_0(\xi) = P_\Sigma([\beta\bullet, i, k]) \cdot P_\Sigma([X, k+1, j])$$
- and if ξ is of the form $\prec 0 \succ_{[X\bullet,i,j]}$:
$$P_0(\xi) = P_\Sigma([X, i, j])$$

where, for any given item I,

$$P_\Sigma(I) = \sum_{\xi' \in \mathcal{D}(I)} P_0(\xi')$$

To simplify notations, for any decomposition ξ of I (i.e. $\xi \in \mathcal{D}(I)$), $P_\Sigma(\xi)$ will denote $P_\Sigma(I)$. Notice that $P_\Sigma(I)$ is in fact the P_0-measure of $\mathcal{D}(I)$. Notice also that $P_\Sigma([X,i,j])$ is nothing else but the inside probability of $X \Rightarrow^* w_i^j$: $P_\Sigma([X,i,j]) = \sum_{d \in \left(X \Rightarrow^* w_i^j\right)} P(d)$.

For any item decomposition ξ, the sampling probability is then defined as $P_s(\xi) = \frac{P_0(\xi)}{P_\Sigma(\xi)}$ which clearly is a probability over $\mathcal{D}(\xi)$.

It is interesting to notice the relation that exists between the above defined sampling probability and the one used by Bod in his sampling scheme (Bod 1998): rephrasing what Bod wrote, a definition for the exact sampling method could be "*a subderivation that has n times as large an* inside *probability as another subderivation should also have n times as large a chance to be chosen as this other subderivation*".

It should also be stressed that P_s can be efficiently computed during the parsing analysis phase using the standard computation of inside probabilities.

Let us now compute the sampling probability of a given derivation d. As each item decomposition is chosen independently (cf the sampling algorithm given in table 1), the sampling probability of a derivation is the product of the sampling probabilities of the selected decompositions: $P_s(d) = \prod_{\xi \in d} P_s(\xi)$. It can then be shown by induction (cf Appendix 1) that $P_s(d)$ is equal to $\frac{P(d)}{P_{\text{DOP}}(w_1^m)}$.

The sampling probability of a parse tree T, $P_s(T)$, being the sum of sampling probabilities of its derivations, we have $P_s(T) = \sum_{d \in T} \frac{P(d)}{P_{\text{DOP}}(w_1^m)} = \frac{P_{\text{DOP}}(T)}{P_{\text{DOP}}(w_1^m)} = P_{\text{DOP}}(T|w_1^m)$.[10] $P_s(T)$ therefore is the conditional DOP-probability of the parse tree T knowing the input sentence, which proves the correctness of the exact sampling scheme.

2.5 Statistical Control for Exact Sampling

The issue of the statistical control of the sampling consists in determining the number of samples that are required to **guarantee** that the most frequent parse tree in the sampled set is in fact the most probable one.

The exact sampling method presented in the previous section can be described as sampling a multinomial random variable, the modalities of which are the different parse trees T_i of the considered input sentence w_1^m, associated with their DOP-probabilities $P_{\text{DOP}}(T_i|w_1^m)$. The sampling control problem therefore corresponds to a classical problem of statistical ordering: finding the most probable modality of a multinomial distribution on the basis of a sample of its realizations.

[10] Indeed, $P_{\text{DOP}}(T, w_1^m) = P_{\text{DOP}}(T)$ as we have $T \Rightarrow^* w_1^m$.

The control method proposed by R. Bod for his sampling scheme in the DOP-MPP framework is an illustration of such a mechanism, in which the error probability (i.e. the probability of choosing the wrong parse tree as MPP) is set to be: $\sum_{i>1}(1-(\sqrt{p_1}-\sqrt{p_i})^2)^n$.[11] However the suggested estimator of the error probability by[12] $\sum_{i>1}(1-(\sqrt{f_{[1]}}-\sqrt{f_{[i]}})^2)^n$ is difficult to justify. The impact on the results of its use instead of the exact value is hard to evaluate.

It is therefore important to consider more sophisticated methods referenced in the statistics literature. We describe hereafter one of these: the Bechhofer-Kiefel-Sobel truncated method (BKST) (Bechhofer and Goldsman 1985), which is a sequential truncated sampling method that combines two other approaches: BEM and BKS sampling.

2.5.1 The BEM, BKS and BKST sampling methods

BEM sampling

It is known (Kesten and Morse 1959) that, for any multinomial random variable with K modalities such as $p_1 \geq \theta\, p_2$ with $\theta > 1$,[13] the probability for the most frequent modality in a sample to effectively be the most probable one is always bigger than the probability $P_{\min}$ of selecting the most probable modality in the case where all the modalities but the most probable one have equal probabilities. This lower bound $P_{\min}$ can be *a priori* computed as a function of K, θ and n (Bechhofer et al. 1959).

The non-sequential BEM sampling method then consists in:

1. defining the tightness of the control by choosing some *a priori* values for θ $(1 < \theta < +\infty)$ and P_{err} $(0 < P_{\text{err}} < 1 - \frac{1}{K})$. θ is intended to be a lower bound for the ratio $\frac{p_1}{p_2}$, which is a quantification of the difficulty of the selection problem.[14]
2. computing the smallest sample size n so that $P_{\text{err}} \leq 1 - P_{\min}(K, \theta, n)$,[15]

[11] n is the number of samples and p_i is the probability of the i-th most probable parse tree.

[12] $f_{[i]}$ is the relative frequency in the sampled set of the i-th most frequent parse tree that set. Notice that $[i]$ might very well be different from i and therefore $f_{[i]}$ is not guaranteed to be a consistent estimator of p_i.

[13] $p_1, ..., p_K$ being the K parameters of the multinomial in decreasing order: $p_1 \geq ... \geq p_K$ (with $p_1 + ... + p_K = 1$).

[14] The bigger $\frac{p_1}{p_2}$, the easier it is to distinguish the most probable modality from the second most probable one.

[15] For large values of n, the following approximation can be used: $n = \left\lceil \frac{\Lambda(K,1-P_{\text{err}})^2(1+\sqrt{\frac{\theta}{(K-1)(\theta+K-2)}})}{4(\arcsin(\sqrt{\frac{\theta}{\theta+K-1}})-\arcsin(\sqrt{\frac{1}{\theta+K-1}}))^2} \right\rceil$, where $\Lambda(K,p)$ is the reverse function of the $(K-1)$-th integral of the $(K-1)$-variate normal density distribution with zero mean, unit variance and uniform correlation 0.5. Tables for $\Lambda(K,p)$ values can be found in (Gupta 1956).

3. generating a set of n independent realizations (sampled set) and selecting the most frequent modality in the sampled set as the most probable one.

BKS sampling

BKS is a sequential sampling method that relies on the following result (Bechhofer et al. 1968): for any multinomial random variable with K modalities such as $p_1 \geq \theta\, p_2$ with $\theta > 1$, the probability for the most frequent modality in a sample to effectively be the most probable one is always bigger than $\frac{1}{1+Z}$ where $Z = \sum_{i=2}^{K} (\frac{1}{\theta})^{n \cdot (f_{[1]} - f_{[i]})}$. The BKS method is then the following:

1. choosing some *a priori* values for θ and P_{err}
2. keeping on sampling modalities, updating[16] the $f_{[i]}$ and Z, as long as $Z > \frac{P_{\text{err}}}{1-P_{\text{err}}}$,
3. selecting as the most probable modality, the most frequent one in the sampled set.

BKST sampling

BKST is a sequential truncated sampling method that combines BEM and BKS: BEM is included in BKS by adding the BEM stopping condition to the BKS stopping condition, which guarantees that BKST is faster than both BEM and BKS.

For all these sampling methods, if p_1 and p_2 are such as $p_1 \geq \theta\, p_2$, then the selected modality is the most probable one, with a probability greater than $1 - P_{\text{err}}$.

For any given sequence w_1^m, BKST can be applied to the MPP problem in the following way:

1. perform the analysis phase of the parsing of w_1^m, i.e. produce the chart containing the compact representation of all the derivations of all the parse trees of w_1^m;
2. compute K, the number of different parse trees for w_1^m. This can either be is done by performing the parsing analysis phase again but with the context-free grammar consisting of only depth-1 elementary trees or even be directly integrated in the original analysis.
3. choose some *a priori* values for θ $(1 < \theta < +\infty)$ and P_{err} $(0 < P_{\text{err}} < 1 - \frac{1}{K})$;
4. compute the smallest sample size $n_{\min}$ so that $P_{\text{err}} \leq 1 - P_{\min}(K, \theta, n_{\min})$
5. use the exact sampling method to extract one derivation from the chart, update the values and the ranks of the relative frequencies $f_{[i]}$ of all the different sampled parse trees, and compute Z;

[16] not only the values but also the order!

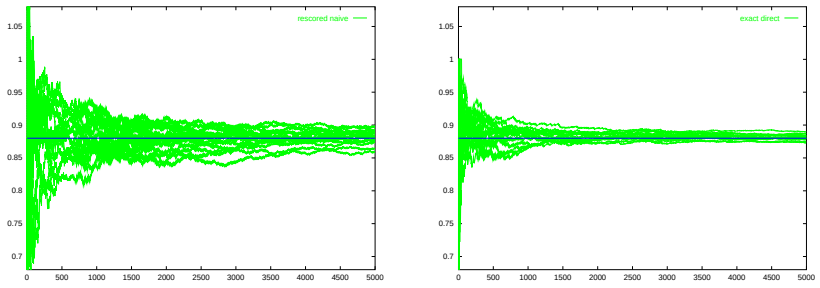


FIGURE 1 (score .vs. sample size) Comparison of convergence of the rescored sampling method (left) and the exact method (right) on 20 runs of 5000 samples converging to the known score of 0.88: the exact method converges much faster.

6. if the total number of sampled derivations is less than $n_{\min}$ and $Z > \frac{P_{\text{err}}}{1-P_{\text{err}}}$, go back to 5;

7. otherwise select as the MPP the most frequent parse tree in the generated sampled set.

2.5.2 Convergence Speed of the Sampling Methods

In addition to the fact that exact sampling enables statistical control, another advantage of this method over rescored sampling is its much faster convergence to the correct distribution of parse trees. This property is illustrated by the simple simulation results given in figure 1, where 20 runs of 5000 samples of a binomial random variable have been produced using both the rescored sampling method and the exact one. This simulation clearly shows that the variance (resp. the convergence rate) is much larger (resp. smaller) for the rescored method than for the exact one.

This fact is easy to understand intuitively: rescoring is needed when the sampling probabilities does not correspond to the correct scores (i.e. DOP-probabilities); the "good" samples are therefore less frequent than they should be if the sampling probability would have been correct. Therefore more samples are needed to have a good approximation of the (correct) MPP.

2.6 Experiments

To experimentally evaluate the performance of the exact sampling method on the DOP model, we have carried out several experiments on Bod's

number of annotated sentences	number of CF rules	number of non-terminals	number of terminals	average sentence length	max. depth of trees
750	381	12	31	8.95	11

TABLE 3 Several characteristics of the corpus used for the experiments.

depth	MPD	MPP fixed	MPP controlled
1	39.4		
≤ 2	46.2	50.1	48.9
≤ 3	48.7	50.2	50.3

TABLE 4 Exact Match Results (in %): "MPP fixed" means MPP estimation using exact sampling with a fixed sample size of 100 derivations per sentence, "MPP controlled" means MPP estimation using BKS-controlled exact sampling with $\theta = 1.5$ and $P_{\text{err}} = 0.05$.

corrected version of the ATIS corpus (Hemphill et al. 1990).[17] The characteristics of that corpus used are given in table 3.

The main objective of our experiments was to verify whether BKST control, besides providing a theoretically grounded control method, also leads to a significant reduction of the sample size.

The evaluation protocol used for the production of the results was the same for all experiments: we compute an average performance on 30 runs consisting each of (randomly) splitting the treebank into a 90% training set and 10% test set; extracting the grammar from the training set[18]; and evaluating performance and other measures on the test set. The performance measure used was the exact tree match.

Concerning the number of derivations for a given input sentence, we obtained an average value of $6.7 \cdot 10^8$ with a standard deviation of $1.1 \cdot 10^9$ (the minimal value was $1,588$ and the maximal one $3,490,106,368$).

Concerning the number of different parse trees for a given input sentence, the minimal value was 13, the maximal value $122,597,062$, and the average value $2.9 \cdot 10^6$ with a standard deviation of $1.6 \cdot 10^7$.

The results indicated in table 4 and table 5 confirm the hypothesis that controlled exact sampling, while producing fully comparable results

[17] Other experiments on the same data can be found in (Bod 1996, Bod and Scha 1996, Goodman 1996, Sima'an 1996, Hoogweg 2000). However, different splits into training and test sets have been used.

[18] the lexicon however was always the same and has been extracted from the whole original corpus; i.e. there were no unknown words in the tests: we did not try to evaluate the performance on Out-of-Vocabulary forms

	number of samples				parse
	min	max	**aver.**	st. dev.	**accur.**
$\theta = 1.2$ $P_{\text{err}} = 0.05$	32	557	**112**	90	**50.3**
$\theta = 1.5$ $P_{\text{err}} = 0.05$	8	1000	**53**	59	**50.3**
$\theta = 2.0$ $P_{\text{err}} = 0.05$	5	408	**30**	29	**50.1**
$\theta = 12.0$ $P_{\text{err}} = 0.05$	2	97	**8**	7	**49.7**
$\theta = 50.0$ $P_{\text{err}} = 0.05$	1	64	**5**	4	**49.5**
$\theta = 1.5$ $P_{\text{err}} = 0.10$	6	1000	**48**	52	**49.3**

TABLE 5 Number of samples for several configurations of the BKS-controlled exact sampling method (depth≤ 3)

with the ones obtained with the standard approach (i.e. fixed sampled size of 100 (Bod 1998)), leads to a substantial reduction of the average number of derivations that need to be sampled.

Notice however that the question of the intrinsic quality of the BKST control method (i.e. the question of knowing how often BKST actually selects the true MPP independently of whether this MPP matches the reference parse tree recorded in the test set) remains open as it would mean the exhaustive exploration of the large parse forest for each sentence, which we did not achieved, due to the huge amount of derivations to run through.

Finally, as θ is one important parameter of the control method, the distribution of the estimated θ values, obtained by computing the $\frac{p_{[1]}}{p_{[2]}}$ ratio, is given in figure 2.

2.7 Conclusions

This chapter presents three important results for the approximation of MPP in the framework of DOP:

1. We have shown that Monte-Carlo sampling techniques can accurately be implemented. We furthermore derive the relationship between (local) sampling probability and the DOP-probability of a sampled parse tree;
2. We have computed the sampling probability of a decomposition that needs to be used so that the sampling probability of a parse tree is exactly equal to its conditional DOP-probability knowing the input sentence. This sampling strategy significantly differs from the former existing ones.
3. Finally we have presented a method that enables control of the sample quality to guarantee sure (with some *a priori* known control error) that the most frequent parse tree in the sampled set is in

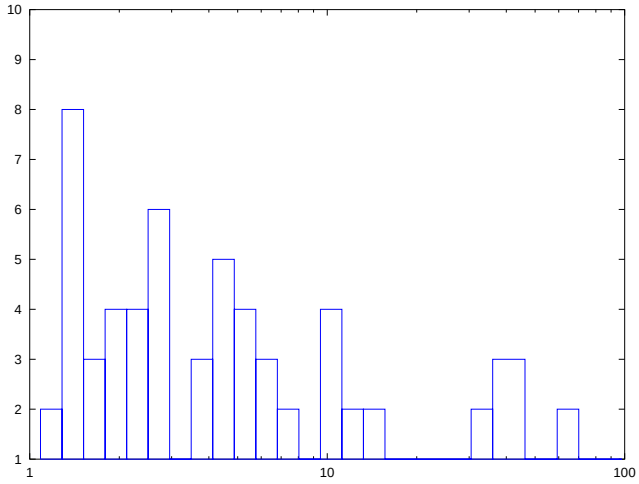


FIGURE 2 Distribution of estimated θ ratio for depth-3 DOP on Bod's corrected version of ATIS corpus. Minimal θ is 1.15, maximal 85, average 12.5 with a standard deviation of 18.6.

fact the MPP. This control method leads to a substantial reduction of the average number of derivations that need to be sampled.

Acknowledgments

The authors would like to thank the editors for their invitation to contribute, the anonymous reviewers for their helpful comments on this chapter, Rens Bod for providing his cleaned-up version of the ATIS corpus, Romain Vinot and Soham Mazumdar for their help in the experiments.

Appendix 1: Sampling using inside probability (proof)

The purpose of this appendix is to prove that the sampling probability $P_s(d)$ of a derivation d of some input sentence w_1^m is equal to $\frac{P(d)}{P_{\mathrm{DOP}}(w_1^m)}$ if the sampling probability used to select the decompositions ξ in $\mathcal{D}(\xi)$ is equal to $\frac{P_0(\xi)}{P_\Sigma(\xi)}$.

Derivations of w_1^m are sampled among all possible derivations of all possibles parse trees for w_1^m by recursively choosing at random item decompositions, starting from $\mathcal{D}([S,1,m])$ (cf table 1, section 2.2.5). Each decomposition being chosen independently, the sampling probability of a derivation d is the product of the sampling probabilities of the decom-

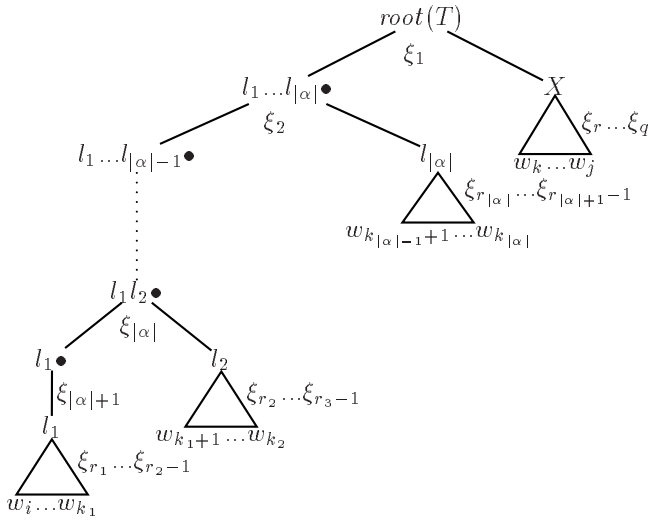


FIGURE 3 Decompositions of a derivation; notations.

positions selected to produce it: $P_s(d) = \prod_{\xi \in d} P_s(\xi)$.

Let us now prove that: $\prod_{\xi \in d} P_s(\xi) = \frac{1}{P_\Sigma(\xi_{\text{root}})} \prod_{t \in d} P(t)$, where ξ_{root} denotes the "*root decomposition* of d", i.e. the decomposition of the top item $[S, 1, m]$ in d.

Let $\Theta(p), p \geq 1$, be the following proposition: any sequence of decompositions $\xi_1, ..., \xi_q$ corresponding to a terminal derivation[19] δ of depth at most p and with ξ_1 as root decomposition, verifies:

$$\prod_{k=1}^{q} P_s(\xi_k) = \frac{1}{P_\Sigma(\xi_1)} \prod_{k=1}^{q} P(\xi_k)$$

Let us prove $\Theta(p)$ by recursion on p.

p = 1 A terminal derivation of depth 1 necessarily corresponds to one single decomposition ξ_1 of the form $\prec t_1, 0 \succ$. Therefore, by definition, $P_0(\xi_1) = P(t_1) = P(\xi_1)$ and $P_s(\xi_1) = \frac{P_0(\xi_1)}{P_\Sigma(\xi_1)} = \frac{1}{P_\Sigma(\xi_1)} P(\xi_1)$.

Recursion Let's assume that $\Theta(p)$ is true. Let $\xi_1, ..., \xi_q$ a sequence of decompositions corresponding to a terminal derivation δ of depth $p+1$ and with $\xi_1 = \prec t, k-1 \succ$ as root decomposition. Let w_i^j be the word sequence derived by δ, $\alpha = \text{Lls}(t) = l_1, ..., l_{|\alpha|}$ and $X = \text{Rl}(t)$ (cf figure 3).

[19] i.e. a derivation the leaves of which are all terminals.

To simplify, and without loss of generality, we assume that

1. the decompositions have been ordered so that $\xi_2, ..., \xi_{|\alpha|+1}$ are respectively the decompositions of the items of the form $[l_1 ... l_{|\alpha|} \bullet, i, k_{|\alpha|}]$, $[l_1 ... l_{|\alpha|-1} \bullet, i, k_{|\alpha|-1}]$, ... $[l_1 \bullet, i, k_1]$ (with $k_{|\alpha|} = k - 1$);
2. all the decompositions ξ_x, $r \leq x \leq q$, for some $r > |\alpha| + 1$, are constitutive of the right-most derivation $X \Rightarrow^* w_k ... w_j$;
3. and each of the terminal derivations $l_x \Rightarrow^* w_{k_{x-1}+1} ... w_{k_x}$, $1 \leq x \leq |\alpha|$ (with $k_0 = i - 1$) corresponds to a sequence of consecutive decompositions $\xi_{r_x} ... \xi_{r_{x+1}-1}$ (with $r_1 = |\alpha| + 2 < r_2 < .. < r_{|\alpha|} < r_{|\alpha|+1} = r$).[20]

With such a convention we have:

$$\xi_1 ... \xi_q = \xi_1 \xi_2 ... \xi_{|\alpha|} \xi_{|\alpha|+1} \xi_{r_1} ... \xi_{r_2-1} \xi_{r_2} ... \xi_{r_{|\alpha|+1}-1} \xi_r ... \xi_q$$

and therefore:

$$\prod_{x=1}^{q} P_s(\xi_x) = P_s(\xi_1) \cdot \left(\prod_{x=2}^{|\alpha|} P_s(\xi_x) \right) \cdot P_s(\xi_{|\alpha|+1}) \cdot \left(\prod_{x=1}^{|\alpha|} \prod_{y=r_x}^{r_{x+1}-1} P_s(\xi_y) \right) \cdot \left(\prod_{x=r}^{q} P_s(\xi_x) \right)$$

which we will now compute piece by piece.

computation of $P_s(\xi_1)$: by definition $P_s(\xi_1) = \frac{P_0(\xi_1)}{P_\Sigma(\xi_1)}$ and $P_0(\xi_1) = P(\xi_1) P_\Sigma(\xi_2) P_\Sigma(\xi_r)$ therefore $P_s(\xi_1) = \frac{1}{P_\Sigma(\xi_1)} P(\xi_1) P_\Sigma(\xi_2) P_\Sigma(\xi_r)$.

computation of $\prod_{x=2}^{|\alpha|} P_s(\xi_x)$: as ξ_x $(2 \leq x \leq |\alpha|)$ is a decomposition of the item $[l_1 ... l_{|\alpha|-x+2} \bullet, i, k_{|\alpha|-x+2}]$, we have

$$P_0(\xi_x) = P_\Sigma(\xi_{x+1}) P_\Sigma([l_{|\alpha|-x+2}, k_{|\alpha|-x+1} + 1, k_{|\alpha|-x+2}])$$

and therefore

$$\begin{aligned}
\prod_{x=2}^{|\alpha|} P_s(\xi_x) &= \prod_{x=2}^{|\alpha|} \frac{P_0(\xi_x)}{P_\Sigma(\xi_x)} = \prod_{x=2}^{|\alpha|} \frac{P_\Sigma(\xi_{x+1})}{P_\Sigma(\xi_x)} P_\Sigma([l_{|\alpha|-x+2}, k_{|\alpha|-x+1} + 1, k_{|\alpha|-x+2}]) \\
&= \frac{P_\Sigma(\xi_{|\alpha|+1})}{P_\Sigma(\xi_2)} \prod_{x=2}^{|\alpha|} P_\Sigma([l_{|\alpha|-x+2}, k_{|\alpha|-x+1} + 1, k_{|\alpha|-x+2}]) \\
&= \frac{P_\Sigma(\xi_{|\alpha|+1})}{P_\Sigma(\xi_2)} \prod_{x=2}^{|\alpha|} P_\Sigma([l_x, k_{x-1} + 1, k_x])
\end{aligned}$$

computation of $P_s(\xi_{|\alpha|+1})$: as $\xi_{|\alpha|+1}$ is a decomposition of $[l_1 \bullet, i, k_1]$, $P_0(\xi_{|\alpha|+1}) = 1$ and $P_\Sigma(\xi_{|\alpha|+1}) = 1$, and therefore $P_s(\xi_{|\alpha|+1}) = 1$.

[20] In the case where l_x is already a terminal, ξ_{r_x} does simply not exist in the sequence.

computation of $\prod_{x=1}^{|\alpha|}\prod_{y=r_x}^{r_{x+1}-1} P_s(\xi_y)$**:** $\xi_{r_x}...\xi_{r_{x+1}-1}$ $(1 \leq x \leq |\alpha|)$ is the sequence of decompositions of the terminal derivation $l_x \Rightarrow^* w_{k_{x-1}+1}...w_{k_x}$, which is a subderivation of δ, and as such is of depth at most p. Therefore, by the recursion hypothesis $\Theta(p)$ we have:

$$\prod_{y=r_x}^{r_{x+1}-1} P_s(\xi_y) = \frac{1}{P_\Sigma(\xi_{r_x})} \prod_{y=r_x}^{r_{x+1}-1} P(\xi_y)$$

Then:

$$\begin{aligned}\prod_{x=1}^{|\alpha|}\prod_{y=r_x}^{r_{x+1}-1} P_s(\xi_y) &= \prod_{x=1}^{|\alpha|} \frac{1}{P_\Sigma(\xi_{r_x})} \prod_{y=r_x}^{r_{x+1}-1} P(\xi_y) = \frac{\prod_{y=|\alpha|+2}^{r-1} P(\xi_y)}{\prod_{x=1}^{|\alpha|} P_\Sigma(\xi_{r_x})} \\ &= \frac{\prod_{y=|\alpha|+2}^{r-1} P(\xi_y)}{\prod_{x=1}^{|\alpha|} P_\Sigma([l_x, k_{x-1}+1, k_x])}\end{aligned}$$

computation of $\prod_{x=r}^{q} P_s(\xi_x)$**:** $\xi_r...\xi_q$ is the sequence of decompositions corresponding to the terminal derivation $X \Rightarrow^* w_{k_1+1}...w_j$, which is a subderivation of d and as such is of depth at most p. Therefore, by the recursion hypothesis $\Theta(p)$ we have:

$$\prod_{x=r}^{q} P_s(\xi_x) = \frac{1}{P_\Sigma(\xi_r)} \prod_{x=r}^{q} P(\xi_x)$$

On the basis of the above mentioned computations, we are now able to compute $\prod_{x=1}^{q} P_s(\xi_x)$:

$$\begin{aligned}
\prod_{x=1}^{q} P_s(\xi_x) &= \left(\frac{1}{P_\Sigma(\xi_1)} P(\xi_1) P_\Sigma(\xi_2) P_\Sigma(\xi_r)\right) \cdot \left(\frac{P_\Sigma(\xi_{|\alpha|+1})}{P_\Sigma(\xi_2)} \prod_{x=2}^{|\alpha|} P_\Sigma([l_x, k_{x-1}+1, k_x])\right) \\
&\quad \cdot \left(\frac{\prod_{y=|\alpha|+2}^{r-1} P(\xi_y)}{\prod_{x=1}^{|\alpha|} P_\Sigma([l_x, k_{x-1}+1, k_x])}\right) \cdot \left(\frac{1}{P_\Sigma(\xi_r)} \prod_{x=r}^{q} P(\xi_x)\right) \\
&= \frac{1}{P_\Sigma(\xi_1)} \cdot P(\xi_1) \prod_{y=|\alpha|+2}^{q} P(\xi_y) \cdot \frac{P_\Sigma(\xi_{|\alpha|+1})}{P_\Sigma([l_1, i, k_1])} \\
&= \frac{1}{P_\Sigma(\xi_1)} \cdot \frac{\prod_{y=1}^{q} P(\xi_y)}{\prod_{y=2}^{|\alpha|+1} P(\xi_y)} \cdot \frac{P_\Sigma(\xi_{|\alpha|+1})}{P_\Sigma([l_1, i, k_1])}
\end{aligned}$$

But, as $\xi_2, ..., \xi_{|\alpha|+1}$ are decompositions of $[l_1 ... l_{|\alpha|}\bullet, i, k_{|\alpha|}], ..., [l_1\bullet, i, k_1]$ respectively, $P(\xi_2) = ... = P(\xi_{|\alpha|+1}) = 1$. Furthermore, by definition, $P_\Sigma([l_1, i, k_1]) = P_0(\xi_{|\alpha|+1})$, and, as $\mathcal{D}([l_1\bullet, i, k])$ reduces to $\left\{\prec 0 \succ_{[l_1\bullet, i, k]} = \xi_{|\alpha|+1}\right\}$, $P_0(\xi_{|\alpha|+1}) = P_\Sigma(\xi_{|\alpha|+1})$; therefore

$$\prod_{x=1}^{q} P_s(\xi_x) = \frac{1}{P_\Sigma(\xi_1)} \prod_{y=1}^{q} P(\xi_y)$$

which shows that $\Theta(p+1)$ holds and ends the demonstration of $\Theta(p)$.□

Finally

$$\prod_{\xi \in d} P_s(\xi) = \frac{1}{P_\Sigma(\xi_{\text{root}})} \prod_{\xi \in d} P(\xi) = \frac{1}{P_\Sigma(\xi_{\text{root}})} \prod_{t \in d} P(t)$$

as by the definitions of the $P(\xi)$ we have $\prod_{\xi \in d} P(\xi) = \prod_{t \in d} P(t)$.

Notice that $P_\Sigma(\xi_{\text{root}}) = P_{\text{DOP}}(w_1^m)$, as by definition $P_\Sigma(\xi_{\text{root}}) = P_\Sigma([S, 1, m]) = \sum_{d' \in \left(S \Rightarrow^* w_1^m\right)} P(d')$.

Therefore for any derivation d of w_1^m starting from S, we have:

$$P_s(d) = \prod_{\xi \in d} P_s(\xi) = \frac{1}{P_\Sigma(\xi_{\text{root}})} \prod_{\xi \in d} P(\xi) = \frac{1}{P_{\text{DOP}}(w_1^m)} \prod_{t \in d} P(t) = \frac{P(d)}{P_{\text{DOP}}(w_1^m)}$$

which concludes the whole demonstration.

References

Bechhofer, R.E., S. Elmaghraby, and N. Morse. 1959. A single-sample multiple-decision procedure for selecting the multinomial event which has the largest probability. *Ann. Math. Statist.* 30:102–119.

Bechhofer, R.E., and D.M. Goldsman. 1985. Truncation of the Bechhofer-Kiefer-Sobel sequential procedure for selecting the multinomial event which has the largest probability. *Communications in Statistics: simulation and computation* 14(2):283–315.

Bechhofer, R.E., J. Kiefer, and M. Sobel. 1968. *Sequential Identification and Ranking Procedures.* University of Chicago Press, Chicago.

Bod, R. 1992. Applying Monte Carlo Techniques to Data Oriented Parsing. In *Proceedings Computational Linguistics in the Netherlands.* Tilburg (The Netherlands).

Bod, R. 1995. *Enriching Linguistics with Statistics: Performance Models of Natural Language.* Amsterdam (The Netherlands): Academische Pers.

Bod, R. 1996. Two questions about Data-Oriented Parsing. In *proc. of 4th Workshop on Very Large Corpora.* Copenhagen.

Bod, R. 1998. *Beyond Grammar, An Experience-Based Theory of Language.* CSLI Lecture Notes, No. 88. Standford (CA): CSLI Publications.

Bod, R., and R. Scha. 1996. Data-Oriented Language Processing: An Overview. Technical Report LP-96-13. Departement of Computational Linguistics, University of Amsterdam. cmp-lg/9611003.

Chappelier, J.-C., and M. Rajman. 1998. A Generalized CYK Algorithm for Parsing Stochastic CFG. In *TAPD'98 Workshop*, 133–137. Paris (France).

Chappelier, J.-C., and M. Rajman. 2000. Monte-Carlo Sampling for NP-Hard Maximization Problems in the Framework of Weighted Parsing. In *Natural Language Processing – NLP 2000*, ed. D.N.

Christodoulakis. 106–117. Lecture Notes in Artificial Intelligence, No. 1835. Springer.

Chappelier, J.-C., and M. Rajman. 2001. Polynomial Tree Substitution Grammars: an efficient framework for Data-Oriented Parsing. In *Proc. of Recent Advances in Natural Language Processing (RANLP-2001)*.

Goodman, J. 1996. Efficient Algorithms for parsing the DOP model. In *Proc. of the Conf. on Empirical Methods in Natural Language Processing*, 143–152. May.

Goodman, J. 1998. *Parsing Inside-Out*. Doctoral dissertation, Harvard University, May. cmp-lg/9805007.

Green, P. J. 1995. Overview: Monte Carlo methods. In *Complex Stochastic Systems and Engineering*, ed. D. M. Titterington, 183–190. IMA Conference Series, No. 54. Oxford Univ. Press.

Gupta, S.S. 1956. *On a decision rule for a problem in ranking means*. Institute of Statistics Mimeograph Series, No. 150. University of North Carolina. May.

Hall, A. 1873. On an experimental determination of PI.

Hammersley, J., and D. Handscomb. 1964. *Monte-Carlo Methods*. Chapman and Hall.

Hemphill, C. T., J; J. Godfrey, and G. R. Doddington. 1990. The ATIS spoken language systems pilot corpus. In *DARPA Speech and Natural Language Workshop*, ed. Morgan Kaufmann. June.

Hoogweg, L. 2000. Extending DOP1 with the Insertion Operation. Master's thesis, University of Amsterdam, January.

Jelinek, F., J. D. Lafferty, and R. L. Mercer. 1992. Basic Methods of Probabilistic Context-Free Grammars. In *Speech Recognition and Understanding: Recent Advances, Trends and Applications*, ed. P. Laface and R. De Mori. F: Computer and System Science, Vol. 75. Springer.

Kesten, H., and N. Morse. 1959. A property of the multinomial distribution. *Ann. Math. Statist.* 30:120–127.

Robert, C., and G. Casella. 1999. *Monte Carlo Statistical Methods*. New York: Springer-Verlag.

Schabes, Yves. 1992. Stochastic Lexicalized Tree-Adjoining Grammars. In *Proc. 14th Int. Conf. of Computationnal Linguistics (COLING)*, 426–432. Nantes (France), August.

Sima'an, K. 1996. Computational Complexity of Probabilistic Disambiguation by means of Tree Grammars. In *Proceedings of COLING'96*. Copenhagen (Denmark). cmp-lg/9606019.

Stolcke, A. 1995. An Efficient Probabilistic Context-Free Parsing Algorithm that Computes Prefix Probabilities. *Computational Linguistics* 21(2):165–201.